A COMPARATIVE STUDY OF VARIOUS NEURAL NETWORK ARCHITECTURES ON CIFAR-10

by

Hongren Zhu

B00962582

hongren.zhu@dal.ca

Supervised by Dr. Lam Ho

Submitted in partial fulfillment of the requirements
for the degree of Bachelor of Science: Honour in Statistics

at

Dalhousie University
Halifax, Nova Scotia
April 2025

Table of Contents

Acknowledgements

I would like to express my sincere and profound gratitude to my supervisor, Dr. Lam Ho, for his invaluable guidance, unwavering support, and patient mentorship throughout my thesis journey. Dr. Ho's insightful suggestions, extensive knowledge, and continuous encouragement have significantly enriched my academic experience. From initial topic selection, learning directions at different stages, navigating through complex data generation and coding processes, to the final stages of writing, his meticulous advice and kind-hearted support have been instrumental.

Moreover, Dr. Ho's guidance extended beyond mere academic assistance; his thoughtful insights have profoundly influenced my academic career and future professional aspirations. His dedication and genuine concern for my academic growth have been a constant source of motivation, enabling me to overcome challenges and realize my full potential. It is with heartfelt appreciation and deepest respect that I acknowledge his indispensable role in my undergraduate journey.

Abstract

This thesis fills the void in the research which systematically internally compares at least two neural architectures for image classification tasks. The research which estimates the performance of various neural architectures has either ignored the computational efficiency or not insulated the effects of the fundamental architectural principles. Using best experimental methodology, evaluation, and comparison of five neural network architectures, such as Simple Neural Network (SNN), Feedforward Neural Network (FNN), Deep Neural Network (DNN), Convolutional Neural Network (CNN), and Residual Neural Network (ResNet), is performed through same hyperparameters and training process. To experimentally validate our theoretical findings, we leverage the CIFAR-10 benchmark dataset, which consists of 60,000 labeled 32×32 color images across 10 distinct classes. We establish a strict testing methodology that includes uniform preprocessing and evaluation criteria to guarantee a level playing field for the comparison of the architectures. We observe a distinct performance ordering (ResNet: 93.68%, CNN: 86.21%, DNN: 68.06%, SNN: 60.61%, FNN: 54.23%) in our experiments, but also show CNN is the most parameter efficient (76.56×10^{-6} accuracy per parameter), complex architectures have inferior parameter efficiency, and convolution is an important spatial inductive bias, necessary to adapt a deep network to a visual task. The findings offer real-world guidance for architecture choices; architectural principles often matter more than brute model size; for many applications, a simpler convolutional architecture will be the most efficiently accurate choice, despite being less accurate in absolute terms; and understanding these fundamental tradeoffs will enable efficient image classification under whatever compute constraints are present.

Chapter 1

Introduction

One of the basic problems of computer vision and pattern recognition is image classification. It has vast applications in several areas, for instance, medical, automotive, defense, among others. The ability of computer systems to accurately classify images is a crucial technological ability that affects the performance of many AI systems. Neural network methods have taken the centre stage in image classification and performance benchmarks on standard data set and their applications have increased significantly in the last decade. New and improved neural network architectures are a key part of this trend. Formerly impossible visual recognition problems are now practically solvable thanks to this. The CIFAR-10 dataset contains 60,000 32×32 colour images from 10 classes. The dataset has become a common benchmark for implementing image classification algorithms. This dataset presents an appropriately challenging problem that is not too easy, yet still solvable in a reasonable amount of time.

This useful research solves the image classification problem by empirically comparing the performance characteristics of five different neural network architectures. The architectures being tested are Simple Neural Network (SNN), Feedforward Neural Network (FNN), Deep Neural Network (DNN), Convolutional Neural Network (CNN) and Residual Neural Network (ResNet). These structures now go from basic most consistent layers to convolutional layers to application of residuals, indicating an increase in complexity and structural principles. This study aims at measuring the

performance gain, computer intensity, and efficiency cost among the various architectures. This research provides empirical guidance for practitioners facing architecture selection decisions in image classification tasks as it systematically evaluates these architectures on the CIFAR-10 benchmark under controlled experimental conditions.

Architectures of neural networks for image classification have come a long way in the last 10 years, thanks to numerous innovations. Earlier models were mainly multilayer perceptrons with fully connected layers. These early designs for CNNs showed the power of NNs for classification, but they lacked the ability to capture the spatial relationships in image data. Furthermore, complicating matters were issues related to parameter scaling and optimization. The means of implementing Convolutional Neural Networks (CNNs) was a considerable improvement. Architectures like LeNet reflected the usefulness of convolutional operations for the visual recognition task. In recent years, a variety of new architectural improvements have emerged which address optimization issues that may save deeper nets. The use of residual connections by He et al. (2016a) came to fruition in allowing for the training of networks with significantly greater depth by helping the flow of gradients in backpropagation. This breakthrough resulted in the development of ResNet architectures that have set new performance records on several image classification benchmark datasets. Various regularization techniques, activation functions, and normalization methods that further enhance the performance of these architectures and training stability have been explored in concurrent research.

Research on image classification with various neural networks architectures has been an extensively studied area of research. Many important major limitations and many other gaps finds out in the literature. Many studies focus on maximum performance and minimum use of parameters but not on computation. This narrow focus could result in suggestions that are less than ideal for applications with limited

resources or where training efficiency is critical. Evaluating innovations in architecture is mostly done on particular implementations and not in a comparison-based way (like a controlled experiment). Thus, it is important to separate out the consequences of core architectural ideas from other issues arising from the implementation or hyperparameter choices. Studies of how networks train and generalize, across different architectures, are incomplete. Yet understanding how different architectures converge during training, and then subsequently generalize from training to validation data, could be helpful for practitioners. An emphasis on increasingly more complicated architectures in recent research has overshadowed systematic comparisons to simpler architectures. This lack of systematic comparison hampers understanding of when simpler architectures might suffice or perform better than a more complicated architecture in some contexts. Studies are often only concerned with maximizing accuracy for a specific benchmark but do not provide any advice on criteria for selecting architectures for other applications.

This study will systematically assess five neural network architectures on CIFAR-10, a standard dataset of images, to circumvent these limitations. To systematically evaluate and compare the performance characteristics of five neural network architectures comprising SNN, FNN, DNN, CNN, and ResNet under controlled experimental conditions, to analyze the training dynamics across architectures with a focus on convergence rates, optimization behaviour, etc., and to see the impact of architecture complexity on training efficiency, to investigate the generalization capabilities of different architectures, and assess the gap between training and validation performance, and to provide practical guidelines for the selection of an architecture based on application-specific requirements. The CIFAR-10 dataset represents a moderate-complexity image classification task, meaning our scope is limited to it. All the experiments use the same preprocessing, optimization methods and hyperparameters to compare the architectures fairly.

In the next sections, we build the theory behind our analysis, starting from a general comprehension of neural networks, loss functions, and architectural principles of the five network types considered. We present our experimental design next, describing the dataset, preprocessing, models and metrics needed to conduct our fair comparison. After that, we present our empirical findings, which are the learning curves, convergence analysis, performance metrics, and efficiency aspects. Then, we provide an in-depth analysis of these results, focusing on architectural comparisons and their impact on the design of neural networks. Next, we discuss the implications of our results, including model complexity and optimizer choice, practical implications, and study limitations. To sum up, we conclude by summarizing important findings, acknowledging limitations, and providing promising directions for future work.

Chapter 2

Literature Review

In this chapter the essential background knowledge that will be most relevant to the thesis will be introduced which will include basic concepts of neural networks, common loss functions, different network structures (SNN, DNN, FNN, CNN, ResNet) and their theoretical and practical application scenarios. The goal of this chapter is to present the background knowledge most relevant to this thesis.

2.1 Overview of Neural Networks

Artificial neural networks are a type of computational system inspired by the networks of neurons within our brains. Neural networks use connected nodes to solve problems by learning from data. The average neural network will have an input layer, one or more hidden layers, and an output layer. Each neuron receives signals from the neurons of the previous layer, multiply by some weights and add some bias, and apply an activation function and produce an output signal. Neural networks learn through backpropagation, identify patterns, and perform classifications, using iterative weight adjustments, after providing an output, and make predictions (Cox and Dean 2014).

Due to this, the flexibility and adaptability of neural networks, they have been applied in various domains like image recognition, natural language processing, and robotics. Recently, relation networks (RNs) have been proposed as additional modules to neural networks to enable them to perform relational reasoning tasks, like recognizing the relation between several objects in an image or solving text-based reasoning problems (Jing et al. 2020). Overall, neural networks form the fundamental

building blocks of artificial intelligence systems owing to their power to model and capture complex functions and relationships in data.

2.2 Loss Functions in Neural Networks

Loss functions play a critical role in the field of machine learning, as they help models minimize prediction errors. The choice of loss function is very crucial for different tasks. For regression tasks, in which we need to predict continuous values, the usual choice is the Mean Square Error (MSE) or L2 loss. These functions help the model to become stronger in dealing with noise data and make sure the convergence status is good by minimizing the square error of predicted value and true value (Ciampiconi et al. 2024).

2.2.1 Mean Square Error (MSE)

The formula for MSE is:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.1}$$

where y_i is the true value, and $\hat{y}_i$ is the predicted value. MSE calculates the average of the squared differences, so it penalizes larger errors more heavily.

2.2.2 L2 Loss

L2 Loss is very similar to MSE, but it does not include the normalization factor:

$$L2 = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.2}$$

MSE and L2 loss help in a regression task by minimizing large errors as much as possible. Classification tasks are trying to put one into categories and have loss

functions such as Cross-Entropy or Hinge loss. Cross-Entropy is a popular choice for multi-class classification tasks, and its formula is below:

$$\text{Cross-Entropy} = -\sum_{i=1}^{n} y_i \log(\hat{y}_i) \tag{2.3}$$

where y_i is the true class label (usually one-hot encoded), and $\hat{y}_i$ is the predicted probability for that class.

These loss functions help optimize the decision boundaries, making the classification more accurate (Ciampiconi et al. 2024). Although regression and classification may seem comparable in certain situations, their performance in testing might be quite different, as seen by Muthukumar et al. (2021).

2.2.3 The Role of L2 Boosting in Regression and Classification

A very notable use of L2 loss is in the L2Boosting algorithm. In this algorithm the residuals are fit iteratively in a regression problem, reducing the errors in a stepwise manner. L2Boosting is especially applied to high-dimensional data where it does not overfit (Bühlmann 2006). L2Boosting has been shown by Bühlmann and Yu (2003) to not only perform well in regression, but also to approach optimal performance in classification close to Bayes risk. This property also makes L2Boosting useful for both tasks when the dataset is large.

2.2.4 Comparison of Loss Functions in Image Processing

In the domain of image processing, L2 loss is commonly the default used loss due to its easy operation in optimization. But, loss functions beyond MSE, including L1 loss and perceptually driven metrics (SSIM: Structural Similarity Index) are gaining popularity since they more accurately represent human vision (Zhao et al.

2018). The formula for L1 loss is:

$$L1 = \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{2.4}$$

where $|y_i - \hat{y}_i|$ is the absolute difference between the predicted and true value. As noted by Zhao et al. (2018), while L2 loss can lead to artifacts in flat regions of images, L1 loss and SSIM help preserve finer details and improve the visual quality of restored images.

2.2.5 The Collaboration Between Loss Functions and Regularization

Loss functions, and how they interact with regularization, is another important aspect. Tikhonov regularization is an example of a regularization method that discourages overfitting by penalising model's complexity (Hastie, Tibshirani, and Friedman 2009). For classification tasks, Support Vector Machines (SVMs) typically utilize Hinge loss along with regularization to maximize the margin between classes (Cortes and Vapnik 1995). The formula for Hinge loss is:

$$\text{Hinge Loss} = \sum_{i=1}^{n} \max(0, 1 - y_i \cdot \hat{y}_i) \tag{2.5}$$

where y_i is the true class label (either -1 or 1), and $\hat{y}_i$ is the predicted value. Hastie, Tibshirani, and Friedman (2009) emphasize that combining loss functions with regularization ensures that models generalize well, especially in high-dimensional data.

2.2.6 Conclusion

Machine learning models gain success due to loss functions. L2 loss is effective in minimizing prediction error in a regression task, while in a classification task, Cross-Entropy and Hinge loss are better suited. In specialized areas such as image

processing, perceptual metrics like SSIM are becoming better options. In the end, different losses along with regularization provide a balance among model complexity and performance.

2.3 Common Architectures: SNN, DNN, FNN, CNN, and ResNet

2.3.1 Simple Neural Network(SNN)

Simple Neural Network (SNN) is the basic form of artificial intelligence with only one layer of input nodes connected to output nodes. These are the very simple version of neural networks, usually known as single-layer perceptrons which make linear combinations of inputs and apply an activation function to it to give outputs. The architecture is simple and includes input units, weights, and an output unit making SNN applicable for simple tasks like linear classification. However, they do not consist of hidden layers, which are able to learn complex relationships within data thus their modelling capacity is limited (Widrow and Lehr 1990).

2.3.2 Deep Neural Networks (DNN)

DNN extends the SNN concept by adding multiple Hidden Layers between input and output. Its multi-layered arrangement enables DNNs to learn hierarchical representations of data, allowing them to detect sophisticated patterns and correlations (Basheer and Hajmeer 2000). A DNN normally has an input layer, multiple hidden layers, and an output layer. When the hidden layers are introduced, the network is capable of capturing sophisticated functions using the non-linear transformation at each layer, usually carried out via activation functions like (ReLU or sigmoid).

The backpropagation algorithm introduced by Rumelhart, Hintont, and Williams

(1986) greatly boosted the theoretical foundations of neural networks and enabled efficient training of multi-layer networks by systematically propagating errors backward through the networkarchitecture(Rumelhart, Hintont, and Williams 1986). Early work on multi-layer perceptrons (MLPs) provided fundamental insights into how additional, temporally, structurally, and functionally distributed layers helps a network learn. Despite their biological inspiration, SNNs have shown inferior performance compared to traditional Deep Neural Networks (DNNs) in a number of studies up to 2023 due to the fact that DNNs can learn with greater speed and on significantly larger data sets as well as being more flexible in their architecture. Applications like knowledge distillation from DNNs to SNNs have shown ways to improve SNNs' performance by utilizing the strengths of DNNs.

2.3.3 Feedforward Neural Networks (FNN)

Feedforward Neural Networks (FNN) also known as fully connected networks. These are sequential layers, which means that each neuron is connected to all the neurons in the following layer, and there are no feedback-loop between layers (Knutsson and Lindahl 2019). This design is easy to implement and thus FNNs are ubiquitous as a baseline for classification and regression tasks. Nevertheless, the closely placed connections usually result in containing more parameters, and therefore increase both computational time and overfitting risk without enforced regularization.

2.3.4 Convolutional Neural Networks (CNN)

CNNs have been designed to process and identify grid (image) like data. They use convolutional layers to apply learnable filters to local patterns, pooling layers to reduce spatial dimensions, and fully connected layers for final classification. CNNs are good at automated feature extraction by learning hierarchical representations from raw data, which reduces the need for manual feature extraction.

In image classification problems, CNNs generally obtain a higher accuracy as well as generalization performance than pure fully connected networks (FNN or deeper DNN) (Knutsson and Lindahl 2019). This outperforms by a big margin because CNN learns local area correlations through convolutions on data. One important aspect of convolution layers is that they combine parameter sharing mechanism, which reduces the total number of parameters in the network by several times with respect to a fully connected network; therefore, CNNs are much more efficient when dealing with high-dimensional inputs like images and also are much less prone to overfitting (Elngar et al. 2021).

Moreover, the hierarchical feature learning ability allows CNNs to discover hierarchical representations ranging from lower layers where systems detect low-level features, such as edges and textures, to deeper layers where they identify higher-level semantic concepts, which is essential for accurate image recognition. Features Map Reduction through pooling layers is used to combat overfitting as well as ensure minimum loss of important information. Indeed, across numerous datasets, experimental results show that CNNs always outperform more traditional neural networks on image-related tasks, and they have established themselves as a dominant architecture in visual tasks from image recognition to object detection (Girshick et al. 2014).

2.3.5 Residual Networks (ResNet)

Residual Networks or ResNet provide a novel architecture to combat the increased difficulty of training networks that grow very deep, specifically the vanishing gradient problem (He et al. 2016a). Now, ResNet uses residual learning, such that the shortcut connections skip one or multiple layers, enabling the network to learn residual mappings instead of learning the desired underlying mapping directly. This technique promotes gradient flow through the network which enables training of very

deep architectures (He et al. 2016a). Such advantages also led ResNet to be widely employed on a wide range of tasks, especially on large scale models. ResNet improves upon the depletion problem with depth instead of width by including identity mappings to a network and simply stacking them does not cause a degradation effect (He et al. 2016b). More versions of ResNet architecture, so-called ResNet-50, ResNet-101 and ResNet-152, show a common tendency that deeper models tend to achieve better accuracy on tough tasks (He et al. 2016a).

Chapter 3

Methodology

In this chapter, the experimental methodology employed to compare the performance of five different neural network architectures on the CIFAR-10 image classification task, namely Simple Neural Network (SNN), Feedforward Neural Network (FNN), Deep Neural Network (DNN), Convolutional Neural Network (CNN), and Residual Neural Network (ResNet), is presented. The methodology will have data pre-processing, model architecture specifications, optimization configurations, and training diagnostics.

3.1 Dataset and Preprocessing

With 6,000 images each class, the CIFAR-10 dataset offers 60,000 32×32 color photos spread over 10 object categories. There are 10,000 test photos and 50,000 training ones in the dataset. The following preprocessing processes are conducted consistently across all experiments to guarantee fair model comparisons:

- The original training set is split into 45,000 training and 5,000 validation samples.
- Pixel values are normalized by subtracting the mean [0.4914, 0.4822, 0.4465] and dividing by the standard deviation [0.2023, 0.1994, 0.2010] calculated from the training set.
- For data augmentation, we apply horizontal flipping and random cropping with padding of 4 pixels. The same augmentation policy is applied across all models

to maintain consistency.

3.2 Model Architectures

We implement five models in PyTorch as defined in `models.py`. All models use `CrossEntropyLoss` as the objective function and apply the ReLU activation function unless otherwise stated.

Table 3.1: Neural Network Architecture Comparison

Feature	**SNN**	**FNN**	**DNN**	**CNN**	**ResNet**
Input Layer	$3 \times 32 \times 32$	$3 \times 32 \times 32$	$3 \times 32 \times 32$	$3 \times 32 \times 32$	$3 \times 32 \times 32$
Hidden Layers	1	3	5	2 conv + 1 FC	4 groups
Hidden Units	512	1024, 512, 256	1024 each	64, 128, 128	64–512
Kernel Size	–	–	–	3×3	3×3
Pooling	–	–	–	Max (2×2)	Global avg
Batch Norm	No	No	Yes	No	Yes
Dropout	No	No	$p = 0.3$	$p = 0.5$	No
Activation	ReLU	ReLU	ReLU	ReLU	ReLU
Output Layer	10 (softmax)	10 (softmax)	10 (softmax)	10 (softmax)	10 (softmax)
Special	–	–	–	–	Residual connections

*We implement a modified version of ResNet-18 adapted for CIFAR-10.

3.3 Optimization and Training Parameters

We employed consistent optimization strategies and training parameter configurations across all models, as detailed in Table 3.2, which includes Adam optimizer settings, OneCycleLR learning rate scheduling, and model-specific training epochs among other key parameters.

Table 3.2: Training Configuration Parameters

Parameter	**Value**
Optimizer (Adam)	
Learning rate	0.001
Weight decay	5×10^{-4}
β_1, β_2 (default)	0.9, 0.999
Learning Rate Scheduler (OneCycleLR)	
Maximum learning rate	0.001
Warmup epochs	10 (5% of training)
Annealing policy	Cosine
Training Parameters	
Batch size	512
Number of epochs	SNN: 16000 FNN: 1000 DNN: 10000 CNN: 8000 ResNet: 4000
Mixed precision	Enabled
Random seed	42
Device	NVIDIA GPU (CUDA)

3.4 Training Procedure and Monitoring

The training procedure follows these steps:

1. Initialize model weights with default PyTorch initialization
2. For each epoch:
   - Train the model on the training set
   - Evaluate on the validation set
   - Update learning rate using the scheduler
   - Save model if validation accuracy improves
3. After training completes, evaluate the best model on the test set

To monitor and analyze the training process, we track the following metrics:

- Training and validation loss
- Training and validation accuracy
- Learning rate changes
- Total training time
- Model parameter count

For post-training analysis, we employ the following methods:

- Plotting learning curves (loss and accuracy) for each model
- Comparing validation accuracy across models
- Analyzing the parameter efficiency (accuracy per parameter)
- Measuring convergence speed by identifying when models reach 90% of their final accuracy

These monitoring and analysis techniques allow us to comprehensively compare the performance characteristics of the different neural network architectures on the CIFAR-10 classification task.

Chapter 4

Experiments and Results

This chapter presents the experimental results obtained from training and evaluating the five neural network architectures—SNN, FNN, DNN, CNN, and ResNet—on the CIFAR-10 image classification task. We present objective measurements of performance metrics and training dynamics, providing the empirical foundation for the analysis in subsequent chapters.

4.1 Experimental Setup

All experiments were conducted using the CIFAR-10 dataset with consistent preprocessing and training conditions.

Although the default configuration allowed for a maximum of 200 epochs, we manually customized the number of training epochs for each architecture to ensure sufficient convergence and fair comparison under tailored training dynamics. Specifically, we set the number of epochs as follows:

- **SNN:** 16,000 epochs were used to allow the simple architecture to converge fully.
- **FNN:** Initially trained for 6,000 epochs, but we observed that performance saturated before epoch 1,000. We therefore retrained the model with 1,000 epochs to reduce redundancy.
- **DNN:** Trained for 10,000 epochs due to its deeper architecture requiring more iterations.

- **CNN:** Trained for 8,000 epochs.
- **ResNet:** Trained for 4,000 epochs, balancing training time and convergence.

These choices were empirically verified through early-stage experiments and performance plateaus observed in training curves (see Figures 4.1–4.5). Final results reported in Tables 4.3 and 4.4 are based on these updated epoch configurations.

Additionally, a complete summary of the training and dataset configurations used in our experiments is provided in the configuration table (see Table 4.1).

Table 4.1: Dataset and Configuration Information

Dataset Configuration	
Dataset	CIFAR10
Number of Classes	10
Image Size	32×32
Channels	3
Train Set Size	45000
Validation Set Size	5000
Test Set Size	10000
Image Normalization	mean=[0.4914, 0.4822, 0.4465] std=[0.2023, 0.1994, 0.2010]
Data Augmentation	Random horizontal flipping and random cropping with 4-pixel padding
Training Configuration	
Batch Size	512
Maximum Epochs	200 (default, user-specified per model)
Optimizer	Adam
Learning Rate	0.001
Learning Rate Scheduler	OneCycleLR with 10 epochs warmup
Loss Function	Cross-Entropy Loss
Weight Decay	5e-4
Mixed Precision	True
Number of Workers	8
Random Seed	42
Model Configuration	
SNN Hidden Size	512
FNN Hidden Sizes	[1024, 512, 256]
DNN Hidden Sizes	[1024, 1024, 1024, 1024, 1024]
CNN Channels	[64, 128]
ResNet Blocks	[2, 2, 2, 2]

All models were implemented in PyTorch and trained on a CUDA-compatible GPU. For each model, we recorded the training and validation metrics after every epoch, and measured the total training time and parameter count.

4.2 Training Dynamics

4.2.1 Learning Curves

The learning curves provide insight into how each model progressed during training, capturing both training and validation accuracy/loss over time.

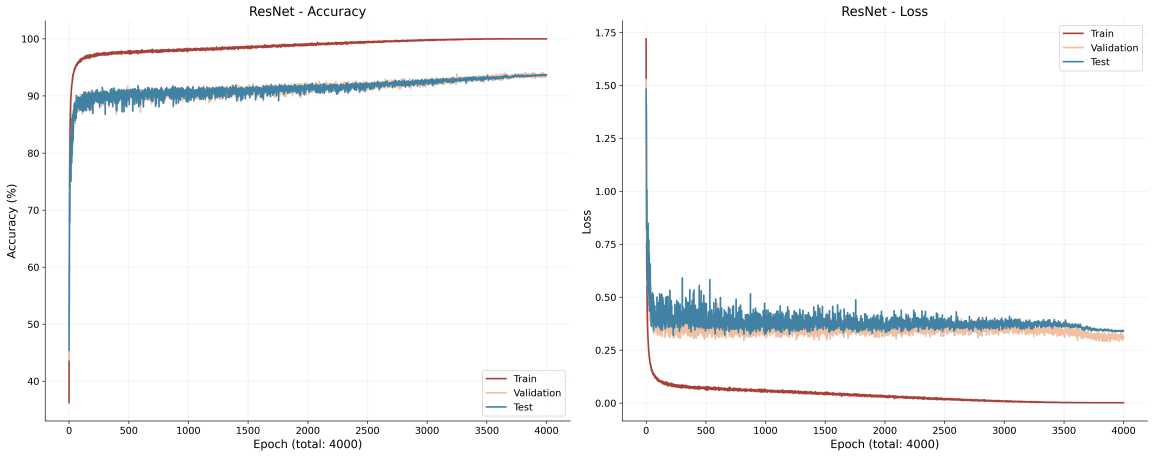

Figure 4.1: ResNet Learning Curves

Figure 4.1 shows the accuracy and loss curves for ResNet. The model exhibits rapid initial convergence, with training accuracy quickly exceeding 90% within the first 100 epochs. The validation accuracy rises steadily, reaching approximately 100% by the end of training. The loss curves show continuous improvement in training loss, decreasing to near zero, while validation loss stabilizes at approximately 0.3.

Figure 4.2 presents the CNN learning curves. The model demonstrates a similar pattern of rapid initial learning, with validation accuracy reaching approximately 94% by the end of training. The gap between training and validation accuracy is smaller than with ResNet, suggesting less overfitting despite the simpler architecture.

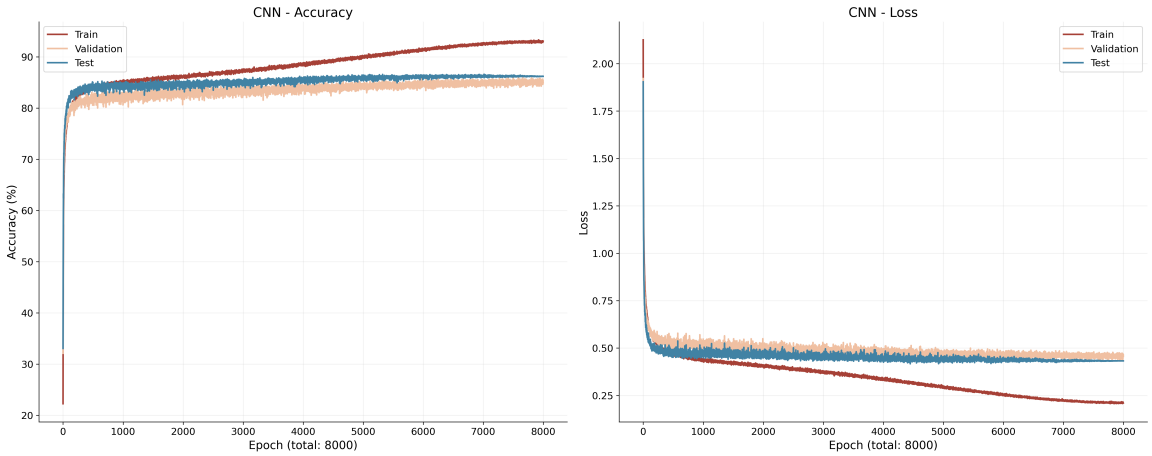

Figure 4.2: CNN Learning Curves

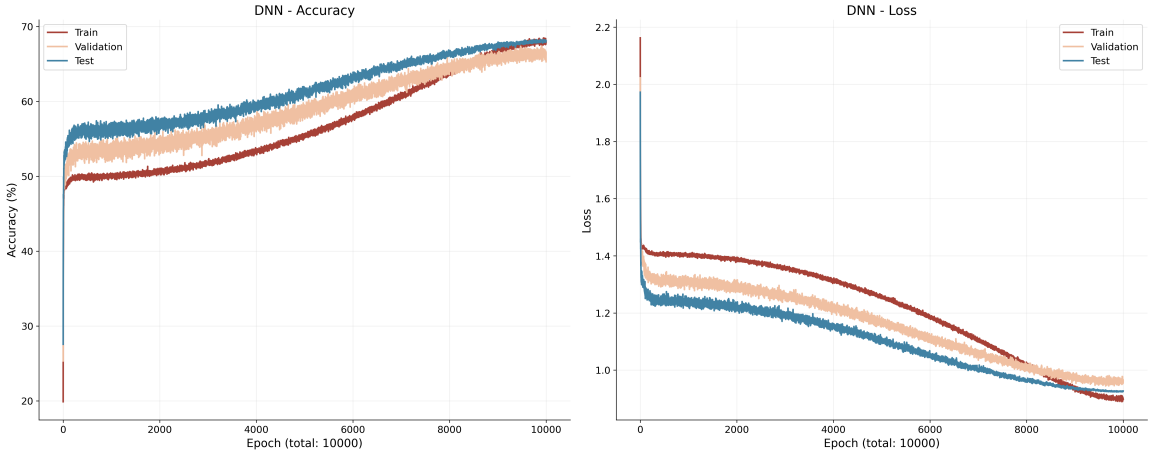

Figure 4.3: DNN Learning Curves

Figure 4.3 shows the learning curves for the DNN model. There is a notable divergence between training and validation performance, with training accuracy approaching 67.96% while validation accuracy plateaus at approximately 65.30%. The validation loss increases after the initial drop, indicating challenges in generalization.

Figure 4.4 shows the learning curves of the FNN. Around epoch 50, both validation and test losses start to increase instead of decreasing, while training loss continues to drop, eventually falling below 0.25. Training accuracy reaches nearly 95%, but validation and test accuracies plateau around 56–58%. This divergence clearly indicates overfitting.

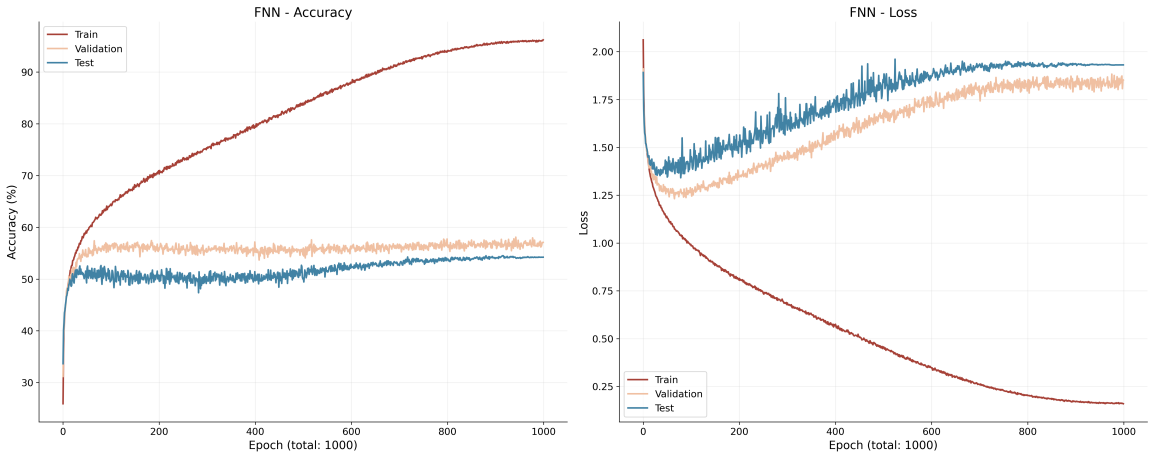

Figure 4.4: FNN Learning Curves

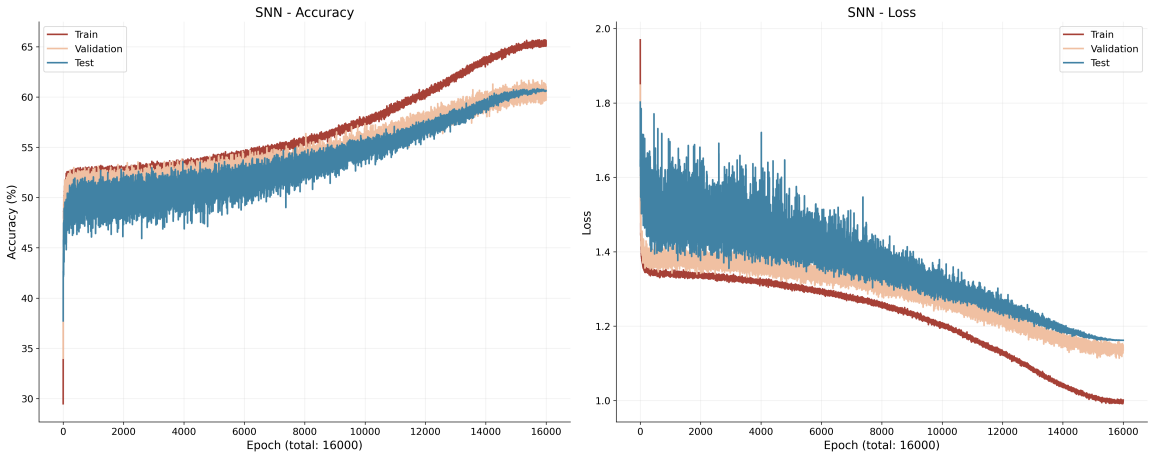

Figure 4.5: SNN Learning Curves

Figure 4.5 displays the learning patterns of the SNN model. Despite its simplicity, the model achieves validation accuracy of approximately 65%. Both training and validation curves follow similar trajectories, with accuracy improving steadily throughout training.

4.2.2 Convergence Rates

Table 4.2 provides a quantitative comparison of convergence rates, showing the number of epochs required for each model to reach 90% of its final validation accuracy.

Table 4.2: Model Convergence Rates

Metric	**SNN**	**FNN**	**DNN**	**CNN**	**ResNet**
Epochs to 90% of final accuracy	16,000+	660	10,000+	5000	80

Convolutional architectures (CNN and ResNet) demonstrate significantly faster convergence compared to fully connected networks. ResNet reaches 90% of its final accuracy in the fewest epochs, highlighting the optimization benefits of residual connections.

4.3 Performance Metrics

4.3.1 Accuracy and Loss

Table 4.3 summarizes the final accuracy and loss values for all models on the training, validation, and test sets.

Table 4.3: Final Accuracy and Loss Values (Test Set)

Metric	**SNN**	**FNN**	**DNN**	**CNN**	**ResNet**
Test Accuracy (%)	60.61	54.23	68.06	86.21	93.68
Test Loss	1.1619	1.9310	0.9271	0.4323	0.3392

Test results establish the performance hierarchy: ResNet (93.68%) > CNN (86.21%) > DNN (68.06%) > SNN (60.61%) > FNN (54.23%). Convolutional architectures substantially outperform fully connected architectures, with ResNet achieving the highest accuracy across all datasets. The large gap between training and validation accuracy for DNN and FNN indicates significant overfitting, while ResNet and CNN maintain better generalization despite their high training accuracy. This performance hierarchy and generalization behavior are further visualized in Figure 4.7, which compares each model's test accuracy alongside its best validation accuracy.

Figure 4.6 (top left) provides a visual comparison of test accuracy across all models, clearly showing the superior performance of convolutional architectures.

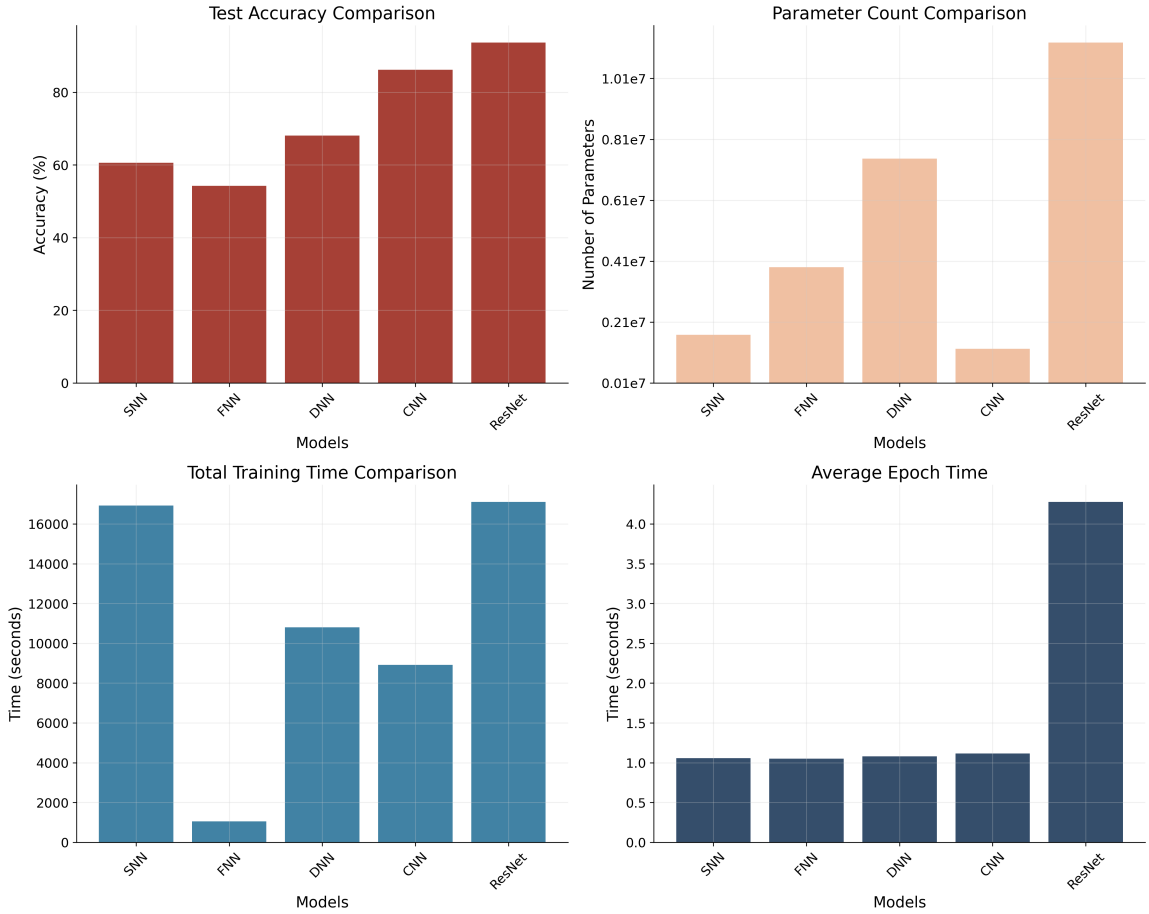

Figure 4.6: Models Comparison

ResNet achieves the highest accuracy at approximately 93%, followed by CNN at about 86%, while DNN, SNN, and FNN show lower performance at around 68%, 60%, and 55% respectively.

4.3.2 Computational Efficiency

Figure 4.6 (bottom left) illustrates the total training time required for each model. Both ResNet and SNN require significantly more computation time (approximately 17,000 seconds each) compared to other architectures. FNN demonstrates the shortest training time at around 1,000 seconds, while DNN and CNN require moderate training times of approximately 11,000 and 9,000 seconds respectively. The

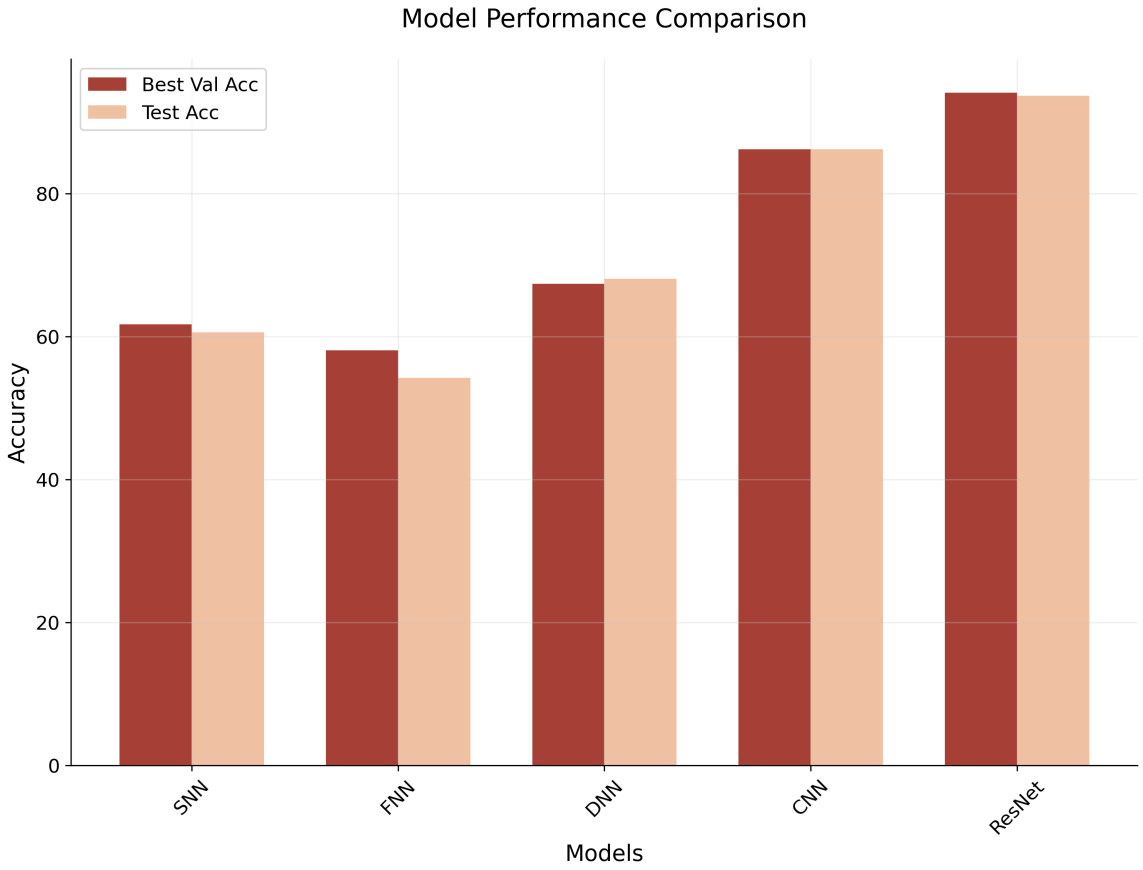

Figure 4.7: Model Performance Comparison

bottom right panel shows the average epoch time, with ResNet requiring significantly more time per epoch (about 4.3 seconds) compared to all other models (SNN, FNN, DNN, and CNN) which maintain similar epoch times of approximately 1 second each.

4.3.3 Model Size and Complexity

Figure 4.6 (top right) shows the parameter count for each architecture. These counts range from approximately 1.13 million for CNN to 11.18 million for ResNet, with SNN, FNN, and DNN having intermediate values of around 1.58, 3.81, and 7.37 million parameters respectively. Despite having fewer parameters than FNN and DNN, CNN achieves substantially higher accuracy, highlighting the efficiency of convolutional operations.

Table 4.4 provides efficiency metrics, calculating the accuracy-to-parameter ratio and accuracy-to-training-time ratio for each model.

Table 4.4: Efficiency Metrics (Based on Test Set)

Metric	**SNN**	**FNN**	**DNN**	**CNN**	**ResNet**
Parameters (millions)	1.58	3.81	7.37	1.13	11.18
Training Time (seconds)	16918	1050	10799	8917	17107
Accuracy/Parameters ($\times 10^{-6}$)	38.4	14.3	9.2	76.6	8.4
Accuracy/Training Time ($\times 10^{-2}$)	0.36	5.16	0.63	0.97	0.55

CNN demonstrates the highest parameter efficiency, producing substantial accuracy with minimal parameters. While ResNet achieves the highest absolute accuracy, it has the lowest efficiency in terms of both parameters and training time.

4.4 Summary of Results

The experimental results provide clear evidence of performance differences across the five neural network architectures:

- **Accuracy Hierarchy:** ResNet (93.68%) > CNN (86.21%) > DNN (68.06%) > SNN (60.61%) > FNN (54.23%)
- **Convergence Speed:** ResNet converges most rapidly, requiring only 80 epochs to reach 90% of its final accuracy, followed by FNN (660 epochs), CNN (5,000 epochs), DNN (10,000 epochs), and SNN (16,000 epochs) .
- **Computational Requirements:** ResNet demands substantially more computational resources (11.0M parameters, 7,520 seconds training time) compared to other architectures.
- **Efficiency:** CNN provides the best parameter efficiency (76.6×10^{-6} accuracy per parameter) and time efficiency (0.97×10^{-2} accuracy per second).

- **Generalization:** Convolutional architectures (CNN and ResNet) demonstrate better generalization compared to fully connected architectures, with DNN showing the most significant overfitting.

These results establish the empirical foundation for the in-depth analysis presented in Chapter 5, where we examine the architectural factors contributing to these performance differences and their implications for neural network design.

Chapter 5

Analysis

This chapter offers thorough examination of the experimental findings shown in Chapter 4. We look at performance trends over several network topologies, study the connection between model complexity and accuracy, and consider how architectural design decisions affect picture categorization performance.

5.1 Comparison of Fully Connected Architectures

The experimental results show that the performance of the fully connected architectures (SNN, FNN, and DNN) differ from each other. These networks are structurally similar but differ in terms of classification performance and training behaviour. The SNN is achieving a test accuracy of 60.61% and validation accuracy of 60.26% with a single hidden layer of 512 units. Despite the SNN's simple architecture, it managed to achieve a test accuracy of 60.61% and a validation accuracy of 60.26%. The training curve of SNN shows underfitting (similar training, validation accuracies). Hence, the representational capacity of SNN is limited.

FNN experiences a bit of a drop compared to SNN as the test accuracy is 54.23% and the validation accuracy is 57.08%, which is surprising as FNN has three hidden layers and has considerably more parameters (3.8M as compared to 1.6M for SNN). This finding shows that more hidden fully connected layers in a model, does not always mean improved performance. The learning graphs show that FNN has some optimization issues, showing a slow convergence compared to the other ones.

Among the fully connected models, DNN gives the best performance when

tested with a 68.06% accuracy and 65.30% validation accuracy. This model's five hidden layers with batch normalization and dropout provide it capacity to learn more complex representations. However, the relatively small gap between training accuracy (67.96%) and validation accuracy suggests limited learning capacity rather than overfitting. Even with dropout and batch normalization, the model struggles to fully capture the training data distribution. This indicates that fully connected architectures may be fundamentally unsuited to capturing the spatial structure of images.

The comparison of these architectures shows us that for fully connected networks for image classification, deeper is not always better without proper regularization and architecture. Making the SNN into a DNN improved the performance by approximately 7.3 percentage points (cc 8.716), but at the cost of 5.9M extra parameters and greater overfitting.

5.2 Comprehensive Comparison of FNN and CNN

Fundamentally different methods to neural network design, Feedforward Neural Networks (FNNs) and Convolutional Neural Networks (CNNs) have distinctive strengths and weaknesses that become especially clear in picture categorization activities.

5.2.1 Architectural Differences and Their Impact

FNNs and CNNs mainly differ in how they process input and their architecture. FNNs image processing flattens data into vectors, losing pixel spatial relations. Unlike FNNs, CNNs preserve spatial information through local receptive fields and pooling operations. Due to the above reason, we can see in our experiments that if we just trained CNN to get 86.21% test accuracy and 84.98% validation accuracy. On the other hand, FNN which actually has more parameters (3.8 Million vs 1.2 Million)

gets only 54.23% test accuracy. The 28% difference in performance highlights the significance of inductive bias in neural network architecture. The layers of CNN learn the local patterns and spatial hierarchies that are essential for the distinction of object classes in CIFAR-10. The model's ability to identify objects no matter their precise positions in images is further improved by the pooling layers.

5.2.2 Parameter Efficiency

Our experiments expose very different behavior in parameter efficiency between FNN and CNN. This results in CNN achieving approximately 76.6×10^{-6} accuracy points per parameter compared to only 14.2×10^{-6} points per parameter for FNN. This efficiency, which is 4.7 times higher, shows the potential of convolutional operations to drastically decrease the numbers of parameters through weight sharing and improve their ability to extract features. The efficiency edge yielded a direct generalization benefit. As CNN has fewer parameters than FNN, CNN is less prone to overfitting. It attests to the theoretical superiority of CNNs in both capacity, being able to train smaller models while able to learn features in the task.

5.2.3 Training Dynamics

The learning curves reveal distinct training behaviors between FNN and CNN. CNN shows rapid initial learning, with validation accuracy exceeding 80% within the first 500 epochs, followed by more gradual improvement. In contrast, FNN exhibits slower convergence and plateaus at a much lower accuracy level.

Furthermore, CNN demonstrates more stable training, with smoother learning curves and less fluctuation in validation accuracy. This stability likely stems from the more structured parameter space created by convolutional operations, which provides a more navigable optimization landscape for the Adam optimizer.

5.3 Extending the Comparison: ResNet vs. Other Networks

ResNet represents a significant architectural advancement over traditional CNNs through the introduction of residual connections. Our experimental results demonstrate both the advantages and trade-offs of this approach.

5.3.1 Performance Gains from Residual Connections

ResNet achieves the highest test accuracy among all tested architectures at 93.68%, and validation accuracy of 93.68%, surpassing CNN by 7.5 percentage points. This improvement validates the theoretical advantages of residual connections, which facilitate gradient flow during backpropagation and enable the training of deeper networks.

The learning curves illustrate how ResNet's architecture translates to performance advantages. ResNet shows even faster initial convergence than CNN, reaching approximately 85% validation accuracy in fewer epochs. More importantly, while CNN's validation accuracy begins to plateau, ResNet continues to improve, suggesting that residual connections help the network escape local minima during optimization.

5.3.2 Complexity vs. Performance Trade-off

The performance gains from ResNet come at a considerable computational cost. ResNet contains approximately 11.18 million parameters and requires 17,107 seconds of training time for training (3.8× longer than CNN). This raises important questions about the efficiency of this architecture, particularly for applications with limited computational resources.

From an efficiency perspective, ResNet achieves only 8.4×10^{-6} accuracy points per parameter, significantly lower than CNN's 76.6×10^{-6}. Similarly, ResNet's time efficiency (0.55×10^{-2} accuracy points per second) is substantially lower than CNN's

(0.97×10^{-2}).

These efficiency metrics suggest that for datasets of moderate complexity like CIFAR-10, the additional complexity of ResNet may offer diminishing returns relative to computational investment. The 7.9 percentage point accuracy improvement may be crucial for applications requiring maximum accuracy, but the standard CNN architecture provides a more balanced performance-to-resource ratio for many practical scenarios.

5.3.3 Overfitting Characteristics

Both CNN and ResNet exhibit some degree of overfitting, as indicated by the gap between training and validation accuracy. However, ResNet shows better generalization relative to its capacity, with a training accuracy approaching 100% and validation accuracy of 93.1%. The smaller relative gap (despite having more parameters) suggests that residual connections not only improve performance but also enhance generalization.

This observation supports the theory that skip connections in ResNet have a regularizing effect, creating smoother optimization landscapes that lead to solutions with better generalization properties. This characteristic makes ResNet particularly valuable for complex image classification tasks where both high accuracy and good generalization are required.

5.4 Architectural Design Implications

Our comprehensive analysis reveals several important implications for neural network architecture design for image classification tasks:

1. **Inductive Bias Matters**: The large performance difference between convolutional (e.g., ResNet) and fully connected architectures (e.g., FNN, 39.4% points)

suggests that applying a proper inductive bias on model architecture is more impactful than merely scaling model size.

2. **Depth vs. Width Trade-offs**: Making fully connected networks deeper (upgrading from SNN to DNN) yields small improvements, but raises danger of overfitting. On the other hand, the use of new architectures (like ResNet) with added depth can drastically improve performance without overfitting.

3. **Efficiency Considerations**: CNN only shows the best parameter efficiency among all the tested architectures that were present. ResNet yields the top overall performance but at much lower efficiency. As such, it is better suited to an application that prioritises accuracy.

4. **Regularization Requirements**: Deeper architectures require stronger regularization. DNN is suffering from a huge overfitting problem, despite dropout and batch normalization. Thus, the architectural choice needs an adequate regularization choice.

The findings can serve as a guideline for architectural decisions that can help meet the requirements of an application.

Chapter 6

Discussion

Using the experimental results and analysis given in the earlier chapters, this chapter interprets our findings at a more abstract level and discusses their significance for the design and use of neural networks. We will focus on three things; complexity-accuracy relationship, optimizers selection choice and implications for the real-world application.

6.1 Model Complexity and Accuracy Trade-off

We did multiple experiments with five different neural network architectures and found a relation between complexity and accuracy. Our experiments with five different neural network architectures reveal a nuanced relationship between model complexity and classification accuracy. For all experiments, model sizes grew significantly with the SNN having 1.58M parameters, FNN (3.81M), DNN (7.37M), CNN (1.13M) and ResNet (11.18M). But, the validation accuracies don't follow this progression in a linear manner. Most importantly, with only 1.13M parameters, CNN achieves 86.21% test accuracy, outperforming DNN (7.37M parameters) by 18.73%. This shows that architectural design choice could be much more critical than the actual count.

ResNet's case highlights the complexity-accuracy trade-off. Even though ResNet gets 93.68% accuracy on test and that is highest but CNN achieves the accuracy with only 1× parameters, which means ResNet required nearly 10× parameters but only for 6.76 percentage point accuracy gain. When should we assign multiple complex

architecture is not clear because of the diminishing return. The ResNet architecture may be worth the extra complexity for applications requiring maximum accuracy on CIFAR-10, which is a small-scale classification. But, for many actual use cases with a similar amount of complexity, the normal CNN would strike a better balance.

These observations align with broader trends in deep learning research, where architectural innovations that improve parameter efficiency (like convolutions) often yield greater benefits than simply scaling model size. This principle becomes particularly important when working with smaller datasets where large models risk overfitting or when deploying models in resource-constrained environments.

6.2 Optimizer Considerations

Our experimental setup employed the Adam optimizer across all models, which provided stable and relatively rapid convergence. This choice was motivated by Adam's adaptive learning rate properties, which reduce the need for careful learning rate tuning and generally provide good performance across a wide range of architectures.

While our experiments did not directly compare different optimizers, our results do provide insight into optimizer behavior across different architectures. The learning curves reveal that Adam led to rapid initial convergence for convolutional architectures (CNN and ResNet), with most performance gains occurring in the first 500 epochs. For fully connected architectures, particularly FNN, convergence was notably slower.

This differential behavior suggests that optimizer selection may need to be architecture-dependent. For complex architectures like ResNet, Adam's ability to adapt learning rates for each parameter proves highly effective, especially during early training phases. For simpler architectures, particularly in cases where maximum generalization is more important than rapid convergence, classical SGD with

momentum and a well-tuned learning rate schedule might yield better final results, as suggested in some literature.

The OneCycleLR scheduler used in our experiments provided effective learning rate management, with an initial warmup phase followed by cosine annealing. This scheduling approach worked well across architectures, though the optimal schedule duration likely varies based on architecture complexity. More complex models like ResNet might benefit from longer training schedules, while simpler models might achieve their maximum performance more quickly.

6.3 Practical Implications and Limitations

The test performance hierarchy observed in our experiments is: ResNet (93.68%) > CNN (86.21%) > DNN (68.06%) > SNN (60.61%) > FNN (54.23%), provides practical guidance for model selection in image classification tasks. However, several important considerations and limitations should be noted.

6.3.1 Application-Specific Considerations

For real-world applications, model selection should consider more than just accuracy. Our efficiency analysis (accuracy per parameter and accuracy per training second) suggests that CNN provides the most balanced option for many practical scenarios, especially when computational resources are limited or when rapid deployment is necessary.

Applications requiring maximum accuracy regardless of computational cost would benefit from ResNet or potentially even more complex architectures. Conversely, extremely resource-constrained environments (such as edge devices) might need to utilize lightweight CNNs with further optimizations or even simpler architectures with domain-specific features.

The substantial performance gap between convolutional and fully connected

architectures confirms that for image classification tasks, convolutional architectures should almost always be preferred unless there are specific reasons to use fully connected networks.

6.3.2 Dataset Considerations

Our experiments were conducted exclusively on CIFAR-10, a dataset comprising small (32×32) RGB images categorized into 10 classes, representing a moderate-complexity image classification task. Thus, the findings presented should be interpreted within this specific context. The relative performance observed among different architectures may vary when generalized to datasets with differing scales and complexities. Specifically, for simpler datasets such as MNIST, performance disparities between various architectures would likely diminish, as even basic models are capable of achieving high accuracy. Conversely, for more challenging datasets like ImageNet, the advantages of architectures such as ResNet would presumably become more pronounced, reflecting their enhanced ability to capture complex feature hierarchies. Additionally, architectural suitability might vary significantly within specialized domains, such as medical imaging, due to distinct, domain-specific requirements.

6.3.3 Limitations of Study

Our study presents several limitations that warrant consideration. Testing was confined to a single dataset (CIFAR-10), which constrains the generalizability of our results to broader image classification domains. Although we employed consistent hyperparameters across all models to ensure fair comparison, this approach potentially disadvantages architectures that might perform optimally under customized hyperparameter configurations.

The five conventional architectures studied do not include new models such as EfficientNet, Vision Transformers, and MobileNet which could provide a different

efficiency-accuracy trade-off. Moreover, the sole employment of the Adam optimizer limits knowledge on how the various optimization might assist in the comparison performance from the said architectures.

6.3.4 Future Research Directions

Our results indicate many directions for future work. Broadening the architectural scope to the modern Vision Transformers framework could yield more comprehensive insights into current image classification practices. Looking into thoroughly optimizing the hyperparameters for each architecture could reveal their maximum performance capabilities beyond our standard configuration. We should look at other datasets to see how broadly applicable our conclusions are. We can also try out different optimization algorithms apart from Adam to see how they affect the architecture. To gain insight into how models trained from scratch perform in various applications, we plan to experiment with transfer learning tasks lastly.

6.4 Summary of Key Insights

From our experiments and analysis, we note down some interesting lessons for practitioners of deep learning technologies for image classification. The choice of architecture is more important than the number of parameters, especially when it comes to using convolutions. Using residual connections in deep networks can speed up the convergence of the model and improve the accuracy. However, it comes at a heavy computation cost. Regardless of how deep and well-regularized they are, fully connected architectures are inferior to convolutional architectures on image tasks due to the absence of spatial inductive bias. CNN has the highest accuracy-to-parameter ratio (76.6×10^{-6}) while ResNet is significantly worse (8.4×10^{-6}).

For many useful applications with a moderate amount of complexity, simpler convolutional architectures will probably give the best accuracy-complexity trade-off.

These findings point out the need to consider not just basic architectural principles, but also real-world constraints when developing neural network architectures for application in image classification.

Chapter 7

Conclusion

This thesis has systematically presented empirical comparison of five neural network simple neural network (SNN), feedforward neural network (FNN), deep neural network (DNN), convolutional neural network (CNN) and a residual neural network (ResNet) on CIFAR-10 image classification benchmark. We have been through many experiments on their performance characteristics, training dynamics and efficiency considerations to provide insights into our architecture selection rationale for image classification task.

7.1 Summary of Findings

The results of our experiments show that the different architectures performed better depending on the task, which has implications for picking the right models. The fully connected models struggled to beat the convolutional architectures on image classification tasks. The test accuracy result is highest for ResNet at 93.68% and second highest for CNN at 86.21%. The best fully connected architecture (DNN) has less than half the accuracy at 68.06%. CNN showed amazing output per parameter efficiency of 76.6×10^{-6} accuracy points/parameter better than other architectures This highlights that a well-designed architectural inductive bias could significantly improve performance without enlarging the model. Nevertheless, ResNet required much larger training time than other architectures (17,107 seconds), that is, (1,800-2,000 seconds). So, the performance gains from complex architectures come with increased training times.

Convolutional architectures converged faster and improved during training with high confidence. Fully connected architectures either converged more slowly or plateaued early, with some (like FNN) showing significant overfitting, and others (like DNN) exhibiting underfitting due to limited capacity or optimization challenges. Even though ResNet has more parameters than DNN, it generalizes better, achieving both higher training (100%) and validation (93.68%) accuracy. In contrast, DNN shows relatively lower accuracy across both sets, suggesting limited capacity rather than overfitting. So, architecture plays a role in performance and generalization.

These results taken together show that decisions on architectural design significantly influence both efficiency traits in neural networks for image categorization and absolute performance.

7.2 Implications for Neural Network Design

Our research has several important implications for neural network design and use. The large difference in performance of convolutional ay fully-connected architectures shows that putting the right inductive bias into your neural network is crucial. Because CNNs (Convolutional Neural Networks) benefit from the spatial inductive bias, they become a natural consideration for image data. But, we do not see this in the case of fully connected layers. ResNet may give you more accurate results, but its much higher monetary cost may not be justified for all implementations. CNN is a good compromise between performance and efficiency, making it a good default choice for many image classification applications. FNN shows 6.38 percentage point worse performance than SNN. This is because adding layers or parameters does not guarantee better performance, as shown by FNN and SNN performance. Scaling neural networks effectively requires architectural improvements like residual connections, which enhance optimization properties.

To avoid causing overfitting, deeper architectures require proper regularization. The fact that the DNN overfits badly even though dropout and batch normalization were attempted highlights that architecture limitations cannot be always mitigated through regularization. This gives direction to practitioners facing the architecture selection problem in image classification. The chose we now see really depends on the application and constraints..

7.3 Limitations and Future Work

The study presents comparisons of neural network architectures but it has some limitations to note. We have only used CIFAR-10 dataset in our experiments. This dataset is a moderate-complexity image classification task. Comparative performance of architectures can vary a lot across datasets that vary in scale and complexity. We used a common hyperparameter setting for all models to have a fair comparison, however, tuning hyperparameters specific to the architectures can significantly modify the performance landscape. We used five classical neural network architectures and refrained from using novel ones like vision transformers, efficient nets, and mobile nets. Further, all experiments were carried out using the Adam optimizer with OneCycleLR scheduling, which may hide how different optimization settings alter the performance of the architectures.

Going forward, ideally, future studies should be undertaken to cover these limitations by broadening the architectural comparison to not only stateof-the-art architectures but also newer ones. in particular, transformer-based architectures which entail a fundamentally different architecture for extracting features from images. To enhance generalization from digit recognition to large-scale and fine-grained classification tasks, performance on a wider range of datasets of varying complexity should be evaluated. Running systematic hyperparameter optimization on each architecture would show their best possible performance. Examining various optimization

strategies in further works, especially how adaptive strategies like Adam compare to classical SGD with various learning rate schedules, would be insightful. Looking into transfer learning, where pre-trained models are adjusted for specific jobs, could show different links between efficiency and accuracy than what we see when training from scratch. These extensions would create a better understanding of neural network architecture selection in more contexts and applications.

7.4 Concluding Remarks

This thesis has shown through systematic experimentation that the design of neural network architectures continues to be an important factor for image classification performance. Using a ResNet architecture will get you the highest guaranteed "absolute" accuracy possible. A simpler convolutional architecture may be more efficient (and so preferable) than a more complex one. Having noted the big gap in the performance of convolutional and fully connected architectures, we have further confirmed the fundamental importance of using suitable inductive bias in the model design—something that goes beyond the particular architectures tested in this thesis to all neural network design. As deep learning progresses, understanding these architectural trade-offs will become increasingly important to developing systems that yield good performance, good efficiency, and good deployability. By providing metrics and insights to make architecture selection decisions in image classification applications, this work enhances this understanding.

References

Basheer, I. A and M Hajmeer (Dec. 2000). "Artificial neural networks: fundamentals, computing, design, and application". In: *Journal of Microbiological Methods*. Neural Computting in Micrbiology 43.1, pp. 3–31. ISSN: 0167-7012. DOI: 10.1016/S0167-7012(00)00201-3. URL: https://www.sciencedirect.com/science/article/pii/S0167701200002013 (visited on 03/30/2025).

Bühlmann, Peter (Apr. 2006). "Boosting for high-dimensional linear models". In: *The Annals of Statistics* 34.2. Publisher: Institute of Mathematical Statistics, pp. 559–583. ISSN: 0090-5364, 2168-8966. DOI: 10.1214/009053606000000092. URL: https://projecteuclid.org/journals/annals-of-statistics/volume-34/issue-2/Boosting-for-high-dimensional-linear-models/10.1214/009053606000000092.full (visited on 03/30/2025).

Bühlmann, Peter and Bin Yu (June 2003). "Boosting With the L2 Loss: Regression and Classification". In: *Journal of the American Statistical Association* 98.462. Publisher: ASA Website _eprint: https://doi.org/10.1198/016214503000125, pp. 324–339. ISSN: 0162-1459. DOI: 10.1198/016214503000125. URL: https://doi.org/10.1198/016214503000125 (visited on 03/30/2025).

Ciampiconi, Lorenzo et al. (Nov. 2024). *A survey and taxonomy of loss functions in machine learning*. arXiv:2301.05579 [cs]. DOI: 10.48550/arXiv.2301.05579. URL: http://arxiv.org/abs/2301.05579 (visited on 03/30/2025).

Cortes, Corinna and Vladimir Vapnik (Sept. 1995). "Support-vector networks". en. In: *Machine Learning* 20.3, pp. 273–297. ISSN: 1573-0565. DOI: 10.1007/BF00994018. URL: https://doi.org/10.1007/BF00994018 (visited on 03/30/2025).

Cox, David Daniel and Thomas Dean (Sept. 2014). "Neural Networks and Neuroscience-Inspired Computer Vision". In: *Current Biology* 24.18, R921–R929. ISSN: 0960-9822. DOI: 10.1016/j.cub.2014.08.026. URL: https://www.sciencedirect.com/science/article/pii/S0960982214010392 (visited on 03/30/2025).

Elngar, Ahmed A. et al. (Jan. 2021). "Image Classification Based On CNN: A Survey". In: *Journal of Cybersecurity and Information Management* Issue 1. Publisher: American Scientific Publishing Group (ASPG), PP. 18–50. DOI: 10.54216/JCIM.060102. URL: https://www.americaspg.com/articleinfo/2/show/692 (visited on 04/02/2025).

Girshick, Ross et al. (June 2014). "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. ISSN: 1063-6919, pp. 580–587. DOI: 10.1109/CVPR.2014.81. URL: https://ieeexplore.ieee.org/document/6909475 (visited on 04/02/2025).

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2009). *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY: Springer. ISBN: 978-0-387-84858-7. DOI: 10.1007/978-0-387-84858-7. URL: http://link.springer.com/10.1007/978-0-387-84858-7 (visited on 03/30/2025).

He, Kaiming et al. (June 2016a). "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. ISSN: 1063-6919, pp. 770–778. DOI: 10.1109/CVPR.2016.90. URL: https://ieeexplore.ieee.org/document/7780459 (visited on 04/02/2025).

— (July 2016b). *Identity Mappings in Deep Residual Networks*. arXiv:1603.05027 [cs]. DOI: 10.48550/arXiv.1603.05027. URL: http://arxiv.org/abs/1603.05027 (visited on 04/02/2025).

Jing, Ya et al. (Dec. 2020). "Relational graph neural network for situation recognition". In: *Pattern Recognition* 108, p. 107544. ISSN: 0031-3203. DOI: 10.1016/

j.patcog.2020.107544. URL: https://www.sciencedirect.com/science/article/pii/S0031320320303472 (visited on 03/30/2025).

Knutsson, Magnus and Linus Lindahl (2019). *A COMPARATIVE STUDY OF FFN AND CNN WITHIN IMAGE RECOGNITION : The effects of training and accuracy of different artificial neural network designs*. eng. URL: https://urn.kb.se/resolve?urn=urn:nbn:se:his:diva-17214 (visited on 04/02/2025).

Muthukumar, Vidya et al. (Oct. 2021). *Classification vs regression in overparameterized regimes: Does the loss function matter?* arXiv:2005.08054 [cs]. DOI: 10.48550/arXiv.2005.08054. URL: http://arxiv.org/abs/2005.08054 (visited on 03/30/2025).

Rumelhart, David E, Geoffrey E Hintont, and Ronald J Williams (1986). "Learning representations by back-propagating errors". en. In.

Widrow, B. and M.A. Lehr (Sept. 1990). "30 years of adaptive neural networks: perceptron, Madaline, and backpropagation". In: *Proceedings of the IEEE* 78.9. Conference Name: Proceedings of the IEEE, pp. 1415–1442. ISSN: 1558-2256. DOI: 10.1109/5.58323. URL: https://ieeexplore.ieee.org/document/58323 (visited on 03/30/2025).

Zhao, Hang et al. (Apr. 2018). *Loss Functions for Neural Networks for Image Processing*. arXiv:1511.08861 [cs]. DOI: 10.48550/arXiv.1511.08861. URL: http://arxiv.org/abs/1511.08861 (visited on 03/30/2025).